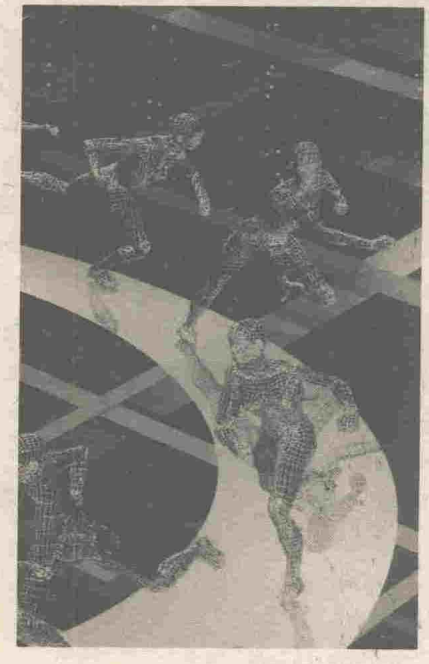
จุฬารัตน์ - ธิติพงศ์ ตันประเสริฐ

ทฤษฎีและการประยุกต์ใช้ โครงข่ายประสาทเทียม ตอนที่ 2



โครงข่ายแพร่ย้อนกลับเป็นแบบจำลองที่ได้รับความนิยมสูงสุดในการนำไปใช้แก้ปัญหา และเป็นแบบจำลองที่นำสาขาโครงข่ายประสาทเทียมกลับมามีชีวิตอีกครั้งหนึ่ง

ก่อนอื่นต้องขออภัยท่านผู้อ่านเป็นอย่างยิ่งในความล่าช้าของตอนที่ 2 ซึ่งห่างจากตอนที่ 1 (ตีพิมพ์ในเล่ม 158) ไปตั้ง 3 เดือน สาเหตุไม่ต้องบอกคงเดากันได้ ก็เหตุยอดนิยมนั่นแหละ คือว่างานเยอะมาก ๆ... เนื่องจากตอนที่ 1 นั้นเล่าไปนานแล้ว ผู้เขียนขอท้าวความเดิมให้ทราบพอให้เห็นภาพโดยสังเขป ในตอนที่ 1 ได้เล่าองค์ประกอบของสมองมนุษย์ เซลล์ประสาทของสิ่งมีชีวิต และนำไปเปรียบเทียบกับเซลล์ประสาทเทียม เพื่อให้เห็นความคล้ายคลึงกัน นอกจากนั้นยังได้แนะนำแบบจำลองทางโครงข่ายประสาทเทียมแบบง่าย ๆ อีกด้วย อันได้แก่ แบบจำลอง McCulloch-Pitts, แบบจำลองเพอร์เซพตรอนเชิงเส้นอย่างง่าย และแบบจำลอง ADALINE กับการเรียนรู้แบบ Widrow-Hoff และการเรียนรู้แบบ LMS

ก่อนที่จะนำเสนอแบบจำลองดัง ๆ ของสาขาโครงข่ายประสาทเทียมต่อไป ผู้เขียนขอเล่าประวัติอย่างย่อของสาขานี้สักนิด เพื่อให้ท่านผู้อ่านเห็นภาพโดยรวมของการถือกำเนิดแบบจำลองต่าง ๆ สักหน่อย เริ่มต้นเมื่อประมาณช่วงทศวรรษ 1940, McCulloch และ Pitts ได้นำเสนอแบบจำลองเซลล์ประสาทเดี่ยว ๆ เป็นครั้งแรก หลังจากนั้นไม่นานในปี 1949, Hebb ก็นำเสนอกฎการเรียนรู้เพื่อใช้อธิบายวิธีการเรียนรู้ของโครงข่ายประสาทเทียม หลังจากนั้นในช่วงประมาณ 20 ปีต่อมาได้มีนักวิจัยนำเสนอรูปแบบจำลองหลาย ๆ แบบ อาทิเช่น Rosenblatt เสนอขบวนการเรียนรู้แบบเพอร์เซพตรอน, Widrow และ Hoff เสนอการเรียนรู้แบบ Widrow-Hoff ขึ้นมาในเวลาใกล้เคียงกัน แต่ในปี 1969 Minsky และ Papert ได้ชี้ให้เห็นข้อจำกัดในการใช้งานของโครงข่ายประสาทเทียมแบบหนึ่งชั้น ซึ่งส่งผลให้งานวิจัยทางสาขานี้เงียบหายไปเป็นระยะเวลาประมาณ 20 ปีทีเดียว กระนั้นก็ตามยังมีนักวิจัยบางท่านคงทำการศึกษาวิจัยต่อมาอย่างไม่หยุดยั้ง มีผลงานที่น่าสนใจเกิดขึ้นบ้าง อาทิเช่น งานของ Anderson (1977), Grossberg (1980), และ Kohonen (1977) เป็นต้น และในปี 1982 เมื่อ Hopfield แนะนำการใช้การลดพลังงานในสาขาฟิสิกส์ให้กับโครงข่ายประสาทเทียม ได้เกิดแรงกระตุ้นในสาขาวิชานี้อีกครั้งหนึ่ง ในปี 1986, Rumelhart และ McClelland ได้เขียนหนังสือชื่อ Parallel Distributed Processing ซึ่งส่งผลให้วงการนี้มีชีวิตขึ้นมาอีกครั้งหนึ่ง พร้อมทั้ง Rumelhart, Hinton และ Williams ได้นำเสนอการเรียนรู้แบบแพร่ย้อนกลับ (Backpropagation Learning Algorithm) ขึ้นด้วย ซึ่งได้รับการพิสูจน์ความสามารถของแบบจำลองด้วยการพัฒนาระบบ NETtalk ขึ้นโดย Sejnowski และ Rosenberg ซึ่งเป็นระบบที่สร้างคำพูดจากอินพุตที่เป็นข้อความภาษาอังกฤษ (Speech Synthesis) การเรียนรู้นี้ได้รับการพัฒนาโดย Werbos (1974) และ Parker (1982) ด้วยโดยอิสระต่อกัน หลังจากนี้ไปมีนักวิจัยหันกลับมาสนใจสาขานี้เป็นจำนวนมาก จนถึงปัจจุบัน มีการนำเสนอรูปแบบจำลองใหม่ ๆ เกิดขึ้นมากมาย

ทฤษฎีและการประยุกต์ใช้
โครงข่ายประสาทเทียม ตอนที่ 2

แบบจำลองโครงข่ายประสาทเทียมเพื่อการระบุกลุ่มรูปแบบ (Artificial Neural Network Models for Pattern Classification)

ในตอนที่ 2 นี้จะขอแนะนำแบบจำลองที่ได้รับความนิยมสูงสุดในด้านขบวนการเรียนรู้แบบมีครูสอน นั่นก็คือโครงข่ายเพอร์เซพตรอนหลายชั้น กับขบวนการเรียนรู้แบบแพร่ย้อนกลับ (Multilayer Perceptron Network with Backpropagation Learning Algorithm) ซึ่งมักจะเรียกกันแบบย่อ ๆ ว่า โครงข่ายแพร่ย้อนกลับ (Backpropagation Network) ซึ่งเป็นแบบจำลองที่นำสาขาโครงข่ายประสาทเทียมกลับมามีชีวิตอีกครั้งหนึ่งนั่นเอง แบบจำลองนี้มีความซับซ้อนมากขึ้นเมื่อเทียบกับแบบจำลองในตอนที่ 1 มีความเป็นทั่วไป (Generalized) มากขึ้น และสามารถใช้งานกับปัญหาที่ไม่เชิงเส้น (nonlinear) ได้

ตำราหลายเล่มจัดหมวดหมู่โครงข่ายประสาทเทียมไว้ค่อนข้างแตกต่างกันพอสมควร แบบที่อาจกล่าวได้ว่ามักถูกใช้บ่อย ๆ ก็เช่น แบบที่แบ่งกลุ่มแบบจำลองของโครงข่ายประสาทเทียมตามลักษณะวิธีการเรียนรู้คือเป็นแบบมีผู้สอน (Supervised) และแบบไม่มีผู้สอน (Unsupervised) และอีกแบบก็คือแบ่งกลุ่มตามลักษณะการใช้งานหรือเรียกได้ว่าแบ่งตามหน้าที่ (Functional Classification) สำหรับในบทความนี้ กล่าวได้ว่าจะใช้ผสมกันทั้งสองแบบ ถ้ามองกันในเชิงลักษณะการใช้งานอาจพอสรุปชนิดของโครงข่ายประสาทเทียมในขอบเขตหนึ่งได้ดังนี้

1. ชนิดระบุกลุ่ม (Classification)
2. ชนิดเกี่ยวข้องกัน (Association)
3. ชนิดจัดตัวเอง (Self-Organization)
4. ชนิดหาค่าที่เหมาะสม (Optimization)

ส่วนการมองโครงข่ายประสาทเทียมตามลักษณะวิธีการเรียนรู้นั้น ในบทความนี้ยึดแนวทางการแบ่งดังนี้

1. แบบปรับค่าน้ำหนักโดยมีผู้สอน (Supervised Learning)
2. แบบปรับค่าน้ำหนักโดยไม่มีผู้สอน (Unsupervised Learning)
3. แบบค่าน้ำหนักไม่เปลี่ยน (Fixed Weight)

แบบจำลองแบบพื้น ๆ ที่สุดที่ใช้ในด้านการระบุกลุ่มคือเพอร์เซพตรอน (Perceptron) ที่ได้กล่าวถึงในตอนที่แล้ว ลักษณะงานระบุกลุ่มคือการให้ผลคำตอบต่อรูปแบบหนึ่ง ๆ ว่ารูปแบบนั้นจัดอยู่ในกลุ่มใด ตัวอย่างเช่น ตัวอักษรตัวหนึ่งอาจมีหลายฟอนต์ แต่ละภาพของตัวอักษรคือรูปแบบที่จะต้องถูกแบ่งกลุ่ม ระบบที่ทำการแบ่งกลุ่มก็จะต้องระบุได้ถูกต้องว่าภาพนั้นเป็นตัวอักษรอะไร (กลุ่มใด) ไม่ว่าภาพนั้นจะเป็นตัวอักษรที่สร้างด้วยฟอนต์ใด

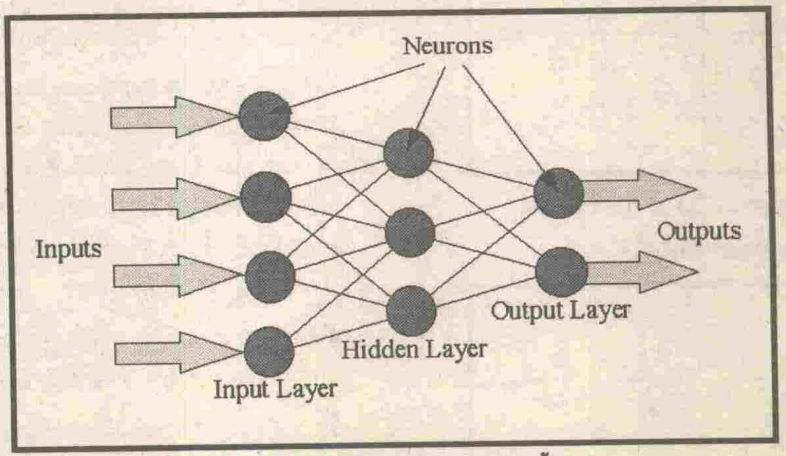
X_1	X_2	X_1 AND X_2
จริง	จริง	จริง
จริง	เท็จ	เท็จ
เท็จ	จริง	เท็จ
เท็จ	เท็จ	เท็จ

ตารางที่ 1 แสดงค่าความจริงของตรรก "และ"

ตรรก "และ" (AND) ซึ่งมีอินพุตสองตัว แต่ละตัวอาจเป็น "จริง" หรือ "เท็จ" และมีผลตอบได้เป็น "จริง" หรือไม่ก็ "เท็จ" ดังตารางที่ 1 จะเห็นได้ว่าถ้าเรามองแต่ละชุดของคู่ (X_1, X_2) เป็นหนึ่งรูปแบบ ในกรณีของตรรก "และ" นี้ เราต้องการระบุได้ว่าแต่ละรูปแบบนั้นอยู่ในกลุ่มที่ให้ผลตอบ "จริง" หรือกลุ่มที่ให้ผลตอบเป็น "เท็จ" แบบจำลองเพอร์เซพตรอนอย่างง่ายที่กล่าวถึงในตอนที่แล้วมีขีดจำกัดคือสามารถใช้งานระบุกลุ่มได้เฉพาะปัญหาที่แบ่งกลุ่มได้ด้วยไฮเปอร์เพลนเชิงเส้นเท่านั้น การจะขยายหลักการของเพอร์เซพตรอนให้สามารถใช้งานได้กับปัญหาไม่เชิงเส้น (nonlinear) นั้นสามารถทำได้โดยการเพิ่มชั้น (layer) ของชุดนิวรอน (neuron) ในการประมวลผล ส่งผลให้เกิดเป็นแบบจำลองเพอร์เซพตรอนหลายชั้น (Multilayered perceptron) ขึ้น

ลองดูสถาปัตยกรรมของโครงข่ายประสาทเทียมที่เรียกได้ว่าเป็นแบบมาตรฐานของเพอร์เซพตรอนหลายชั้นในรูปที่ 1 สถาปัตยกรรมมาตรฐานนี้มีลักษณะหลัก ๆ คือ (สมมติเรียกชั้นอินพุตว่า X ชั้นซ่อนว่า Y และชั้นเอาต์พุตว่า Z)

1. มีชั้นของนิวรอนแทรกระหว่างชั้นอินพุต (X) และชั้นเอาต์พุต (Z) ซึ่งเรียกว่าเป็นชั้นซ่อน (Hidden Layer)



รูปที่ 1 แสดงสถาปัตยกรรมของเพอร์เซพตรอนแบบหลายชั้น (แบบมีชั้นซ่อนชั้นเดียว) (รูปจาก http://www.brainstorm.co.uk/NCTT/tech/tby.htm)

ทฤษฎีและการประยุกต์ใช้ โครงข่ายประสาทเทียม ตอนที่ 2

2. การเชื่อมต่อระหว่างชั้นที่ติดกันเป็นแบบ Fully Connected Feedforward นั่นคือ ทุก ๆ นิวรอนในชั้นอินพุตจะส่งสัญญาณไปยังทุก ๆ นิวรอนในชั้นซ่อนชั้นแรก และทุก ๆ นิวรอนในชั้นซ่อนชั้นแรกจะส่งสัญญาณไปยังทุก ๆ นิวรอนในชั้นต่อ ๆ ไป จนในที่สุดทุก ๆ นิวรอนในชั้นซ่อนชั้นสุดท้ายจะส่งสัญญาณไปยังทุก ๆ นิวรอนในชั้นเอาต์พุต

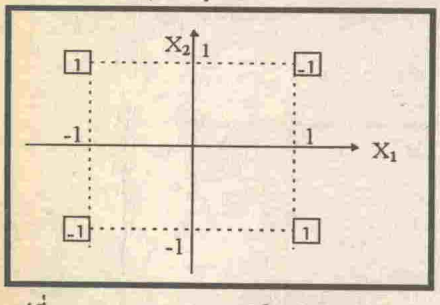
3. ชั้นอินพุตไม่มีการประมวลผล คือทำหน้าที่ในการรับสัญญาณเข้า แล้วกระจายออกไปยังแต่ละนิวรอนในชั้นต่อไปเท่านั้น ส่วนชั้นซ่อนและชั้นเอาต์พุตนั้นมีการประมวลผล

ลองพิจารณาปัญหาไม่เชิงเส้นมาตรฐานของวงการโครงข่ายประสาทเทียม นั่นคือปัญหาตรรก "XOR" แสดงในตารางที่ 2

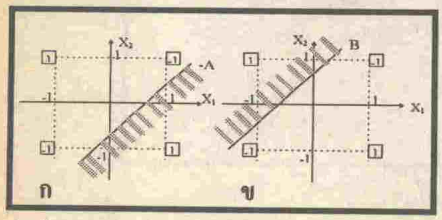
ถ้าเราแทนค่า "จริง" ด้วย 1 และ "เท็จ" ด้วย -1 ตรรก "XOR" นี้สามารถแสดงในเชิงเรขาคณิตวิเคราะห์ได้ดังรูปที่ 2

X1	X2	Z
จริง	จริง	เท็จ
จริง	เท็จ	จริง
เท็จ	จริง	จริง
เท็จ	เท็จ	เท็จ

ตารางที่ 2 แสดงค่าความจริงของตรรก "XOR"



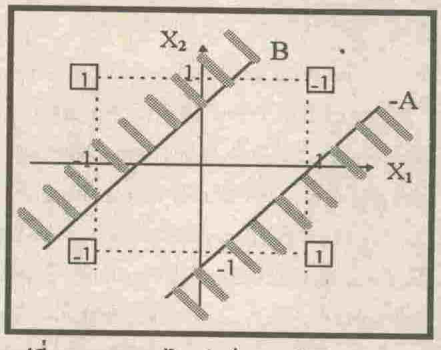
รูปที่ 2 แสดงตรรก "XOR" ในเชิงเรขาคณิตวิเคราะห์



รูปที่ 3

จะเห็นว่าตรรก "XOR" เป็นปัญหาชนิดที่ไม่สามารถแบ่งได้ด้วยไฮเปอร์เพลนเชิงเส้น อย่างไรก็ตาม จะสังเกตได้ว่า ถ้าเราสามารถใช้ไฮเปอร์เพลนสองไฮเปอร์เพลนทำงานร่วมกัน จะสามารถใช้แก้ปัญหานี้ได้ ดังแสดงในรูปที่ 3ก และ 3ข

ถ้าเรานำไฮเปอร์เพลนทั้งสองมารวมกัน จะได้ผลดังรูปที่ 4 ซึ่งทำให้สามารถแบ่งกลุ่มชุดข้อมูล (X_1, X_2) แต่ละชุดได้ถูกต้อง



รูปที่ 4 แสดงสองไฮเปอร์เพลนกับการแก้ปัญหา XOR

อัลกอริธึมของการเรียนรู้แบบแพร่ย้อนกลับ (เฉพาะกรณีมีชั้นซ่อนชั้นเดียว)

ตัวแปรต่าง ๆ ที่ใช้คือ

x_i : อินพุตบิตที่ i สมมติว่ามีอินพุตทั้งหมด n บิต

y_j : ค่ากระตุ้น (activation) ของบิตที่ j ของชั้นซ่อน สมมติว่ามีทั้งหมด m บิต

z_k : เอาต์พุตบิตที่ k สมมติว่ามีทั้งหมด p บิต

α : อัตราการเรียนรู้ มีค่าในช่วง (0, 1]

ขั้นตอนการเรียนรู้แบบการแพร่ย้อนกลับ

1. สร้างโครงข่ายประสาทเทียมให้มีจำนวนอินพุตและเอาต์พุตเท่ากับจำนวนอินพุตบิตและเอาต์พุตบิตของปัญหาที่เราต้องการแก้ไขตามลำดับ ส่วนจำนวนนิวรอนของชั้นซ่อนให้เราเป็นผู้กำหนดเองซึ่งควรจะเริ่มที่ค่ามากพอสมควรก่อน

2. กำหนดค่าเริ่มต้นแก่ค่าน้ำหนักทุกเส้นในโครงข่ายประสาทเทียม ซึ่งควรจะเป็นค่าสุ่มในช่วง (-1,1)

3. คำนวณค่า $\sum_{i=1}^{i=n} x_i w_{ij} = u_j$ สำหรับแต่ละนิวรอน j ของชั้นซ่อน

4. คำนวณค่ากระตุ้น (activation) สำหรับแต่ละนิวรอน j ของชั้นซ่อน : $y_j = f(u_j)$ โดยฟังก์ชันที่นิยมใช้คือ $f(x) = \frac{1}{1+e^{-x}}$ (Logistic function)

5. คำนวณค่า $\sum_{j=1}^{j=m} y_j w_{jk} = v_k$ สำหรับแต่ละนิวรอน k ของชั้นเอาต์พุต

6. คำนวณค่ากระตุ้น (activation) สำหรับแต่ละนิวรอน k ของชั้นเอาต์พุต : $z_k = f(v_k)$

7. นำค่า z_k มาเปรียบเทียบกับค่าเอาต์พุตจริงที่น่าจะเป็น และคำนวณหาค่าความผิดพลาดที่เกิดขึ้น ถ้าค่าความผิดพลาดที่ได้น้อยกว่าระดับที่เราตั้งไว้ ก็จบการเรียนรู้ มิฉะนั้นทำข้อ 8 ต่อ

8. คำนวณค่าของการปรับน้ำหนักสำหรับแต่ละค่าน้ำหนักดังนี้

- สำหรับเส้นเชื่อมระหว่างชั้นซ่อนกับชั้นเอาต์พุต :

$$\Delta w_{jk} = \alpha \delta_k y_j \quad \text{โดยที่} \quad \delta_k = (t_k - z_k) f'(v_k)$$

- สำหรับเส้นเชื่อมระหว่างชั้นอินพุตกับชั้นซ่อน :

$$\Delta w_{ij} = \alpha \delta_j x_i \quad \text{โดยที่} \quad \delta_j = \sum_{k=1}^{p} \delta_k w_{jk} f'(u_j)$$

9. ปรับค่าน้ำหนักของเส้นเชื่อมจากนิวรอน r ไปยังนิวรอน s ด้วยสมการต่อไปนี้

$$w_{rs}(new) = w_{rs}(old) + \Delta w_{rs}$$

10. กลับไปทำข้อ 3.

ทฤษฎีและการประยุกต์ใช้ โครงข่ายประสาทเทียม ตอนที่ 2

การรวมไฮเปอร์เพลนดังกล่าวสามารถมองกลับไปเป็นเพอร์เซพตรอนหลายชั้นได้ดังรูปที่ 5 นั่นคือแต่ละนิวรอนในชั้นซ่อน (hidden layer) จะทำแบบเพอร์เซพตรอนอย่างง่ายคือแบ่งชุดข้อมูลด้วยไฮเปอร์เพลนเชิงเส้น ส่วนเอาต์พุต (นิวรอน Z) ก็จะรวมผลของนิวรอน A และ B ในชั้นซ่อนเพื่อให้ผลการแบ่งกลุ่มที่ไม่เชิงเส้นได้

เมื่อทราบแนวทางของเพอร์เซพตรอนหลายชั้นไปแล้ว ก็เหลือสิ่งสำคัญที่ต้องพิจารณาอีกหนึ่งอย่างคือวิธีการปรับค่าน้ำหนักต่าง ๆ ในเพอร์เซพตรอนหลายชั้นให้ถูกต้อง อัลกอริธึมการเรียนรู้ (Learning Algorithm) ที่เรียกได้ว่ามาคู่กับเพอร์เซพตรอนหลายชั้นคือ "การแพร่ย้อนกลับ" (BackPropagation) ซึ่งโด่งดังที่สุดในวงการโครงข่ายประสาทเทียม

ที่มาของชื่อการแพร่ย้อนกลับนั้นมาจากจุดที่ว่า วิธีการปรับน้ำหนักเพื่อให้ได้ค่าที่เหมาะสมนั้นจะใช้วิธีสอนว่า "ค่าเป้าหมาย" ของแต่ละอินพุตนั้นคืออะไร และใช้ความผิดพลาด (error) ของเอาต์พุตเปรียบเทียบกับ "ค่าเป้าหมาย" เพื่อเป็นตัวชี้นำในการปรับน้ำหนัก ดังนั้น "การแพร่ย้อนกลับ" จึงเป็นขบวนการเรียนรู้ชนิดมีผู้สอน (Supervised) แต่ทว่าเราไม่มี "ค่าเป้าหมาย" ของสัญญาณที่ออกมาจากแต่ละนิวรอนในชั้นซ่อน (เรามีแต่ค่าเป้าหมายของนิวรอนในชั้นเอาต์พุต) ดังนั้นการแก้ไขค่าน้ำหนักสำหรับชั้นซ่อนจึงใช้วิธีแพร่ค่าความผิดพลาดจากชั้นเอาต์พุตกลับมายังชั้นซ่อน

ในแง่มุมหนึ่ง สามารถมองได้ว่าค่าของตัวแปร δ สำหรับชั้นเอาต์พุตก็คือสัดส่วนหนึ่งของความผิดพลาดระหว่างเอาต์พุตของแต่ละนิวรอนกับค่าเป้าหมาย และสำหรับชั้นซ่อนก็คือการรวมกันของสัดส่วนของความผิดพลาดที่แพร่ย้อนกลับมาจากชั้นเอาต์พุตมานั่นเอง

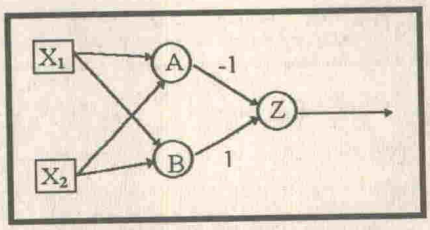
การคำนวณค่ากระตุ้นของแต่ละนิวรอนนั้นไม่ใช้ฟังก์ชันขั้นบันได (step function) เหมือนกับที่ใช้ในเพอร์เซพตรอนอย่างง่ายที่กล่าวถึงในตอนที่แล้ว แต่เป็นฟังก์ชันลอจิสติก (Logistic) สาเหตุมาจากทฤษฎีของการแพร่ย้อนกลับซึ่งต้องใช้ฟังก์ชันที่หาอนุพันธ์ได้สำหรับทุก ๆ ค่าของโดเมน เนื่องจากการปรับค่าน้ำหนักนั้นเป็นสัดส่วนโดยตรงกับเกรเดียนต์ (gradient) ของผลรวมความผิดพลาดระหว่างเอาต์พุตของโครงข่ายประสาทเทียมกับค่าเป้าหมายที่แต่ละชุดของค่าน้ำหนัก ทั้งนี้ฟังก์ชันลอจิสติก (Logistic) มีลักษณะใกล้เคียงกับฟังก์ชันขั้นบันไดและมีคุณสมบัติตามที่ระบุดังแสดงในรูปที่ 6

นอกจากนั้นอนุพันธ์ของฟังก์ชันลอจิสติกยังคำนวณได้ง่ายอีกด้วย เนื่องจากสามารถแสดงให้อยู่ในรูปนิพจน์ของตัวฟังก์ชันลอจิสติกเองได้ นั่นคือ $f'(x) = f(x)[1 - f(x)]$

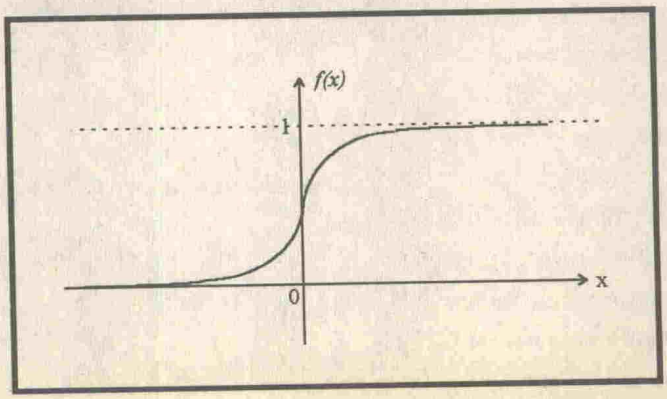
มีทฤษฎีที่กล่าวไว้ว่าชั้นซ่อนเพียงชั้นเดียวก็เพียงพอแล้วสำหรับการจำลองปัญหาไม่เชิงเส้นใด ๆ ก็ได้ แต่ในบางกรณีการเพิ่มชั้นซ่อนก็สามารถช่วยให้การเรียนรู้ง่ายขึ้น

สิ่งที่ตามมากับเพอร์เซพตรอนหลายชั้นและ "การแพร่ย้อนกลับ" คือเราจะรู้ได้อย่างไรว่าควรใช้นิวรอนในชั้นซ่อนกี่นิวรอน คำตอบนี้ยังไม่มีผู้สามารถยืดอกตอบได้อย่างมั่นใจ เนื่องจากจำนวนนิวรอนในชั้นซ่อนนี้ขึ้นอยู่กับความซับซ้อนของปัญหา มีหลายเทคนิคถูกนำเสนอออกมาเพื่อใช้หาจำนวนนิวรอนที่เหมาะสมในชั้นซ่อนนี้ ซึ่งวิธีที่นับว่าแพร่หลายพอสมควรคือใช้นิวรอนให้เยอะ ๆ ไว้ก่อนแล้วค่อย ๆ ลดลงมาจนได้จำนวนที่น้อยที่สุดที่โครงข่ายประสาทเทียมยังคงเรียนรู้ได้ [6] เทคนิคดังกล่าวเรียกว่า พรุนนิง (Pruning)

ปัญหาที่อยู่คู่กับการใช้งานเพอร์เซพตรอนอีกปัญหาหนึ่งคือเราจะรู้ได้อย่างไรว่าตัวอย่างที่เราใช้สอนไปนั้นดีเพียงพอที่จะเป็นตัวแทนข้อมูลทั้งหมด จนสามารถทำให้เราได้คำตอบที่ถูกต้องเป็นอัตราที่สูงจนน่าพอใจในขั้นทดสอบด้วยข้อมูลที่ไม่เคยเห็น (test with unseen data) วิธีที่ว่ากันดื้อ ๆ ก็คือใช้ตัวอย่างให้เยอะ ๆ เข้าไว้ โดยหวังว่าเมื่อกลุ่มตัวอย่างมีขนาดใหญ่ ๆ ก็คงพอจะเป็นตัวแทนของประชากรได้ถูกต้อง อย่างไรก็ตามเป็นที่รู้กันว่ายิ่งโครงข่ายประสาทเทียมมีขนาดใหญ่ขึ้นเท่าไร ก็ยิ่งต้องการจำนวนตัวอย่างที่จะใช้สอนมากขึ้นเป็นทวีคูณ เพื่อให้ได้ผลการทดสอบกับข้อมูลที่ไม่เคยเห็นได้ดีพอ มีหลักการว่าจำนวนตัวอย่างที่ใช้สอนนั้นควรเพิ่มตามจำนวนตัวแปรของระบบซึ่งคือจำนวนค่าน้ำหนัก จำนวนค่าน้ำหนักนี้เพิ่มหรือลดตามจำนวนนิวรอนในแต่ละชั้นนั่นเอง ดังนั้นวิธีหนึ่งซึ่งอาจทำได้ง่ายกว่าการพยายามเพิ่มจำนวนตัวอย่างคือ การพยายามแปลงลักษณะตัวอย่างเพื่อลดขนาด (จำนวนบิต) ของแต่ละตัวอย่างลง ตัวอย่างงานวิจัยเช่น Light Sensitive Model [5] (โดยพยายามคงไว้เฉพาะลักษณะที่จำเป็นจริง ๆ ในการให้ความหมายของข้อมูลตัวอย่างนั้น) ก็สามารถเพิ่มความถูกต้องในการใช้งานโครงข่ายประสาทเทียมได้โดยไม่ต้องใช้ตัวอย่างในการสอนมากเกินไป



รูปที่ 5 แสดงไฮเปอร์เพลนเชิงเรขาคณิตวิเคราะห์ในการแก้ปัญหา "XOR"



รูปที่ 6 ฟังก์ชันลอจิสติก (Logistic Function)

เรื่องน่าอ่าน

ทฤษฎีและการประยุกต์ใช้ โครงข่ายประสาทเทียม ตอนที่ 2

ในฉบับนี้ผู้เขียนจะขอแนะนำซอฟต์แวร์ตัวจำลองโครงข่ายประสาทเทียมของคนไทยให้ได้รู้จักกัน พร้อมทั้งตัวอย่างเพื่อประกอบแสดงวิธีการใช้งานให้ด้วย ซอฟต์แวร์นี้ชื่อว่า Stat Net พัฒนาโดยนักศึกษาภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยอัสสัมชัญ จำนวน 4 ท่าน ได้แก่ นายนิติธรรม โกวิทกูลไกร นายนันทวุฒ ชูกิต นางสาวธีราพร ตียะปัญจนิตย์ และนางสาววรลักษณ์ พรโสภณ ซึ่งโครงการพัฒนาซอฟต์แวร์นี้ได้รับทุนอุดหนุนจากโครงการวิจัย พัฒนาและวิศวกรรม โครงการการพัฒนาซอฟต์แวร์ขนาดเล็ก ปี 2540 จัดโดยศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ หรือเนคเทคนั่นเอง (ผู้สนใจเข้าดูรายละเอียดเพิ่มเติมได้ที่ http://tpt.nectec.or.th/ ซึ่งเป็นโฮมเพจของฝ่ายไตรภาคีของเนคเทค เป็นคณะทำงานดำเนินการโครงการพัฒนาซอฟต์แวร์ขนาดเล็ก ทราบมาว่าโฮมเพจของโครงการเองกำลังจะคลอดเร็ว ๆ นี้ ติดตามรายละเอียดได้จากโฮมเพจของหน้าไตรภาคีนี่แหละ) ซึ่งผลการประกวดปรากฏว่าซอฟต์แวร์นี้ได้รางวัลรองชนะเลิศอันดับ 1 ประเภทซอฟต์แวร์สำหรับงานประยุกต์ (Application Software) เพื่อเป็นการสนับสนุนสินค้าไทยจะขอใช้ตัวจำลอง Stat Net ในการแสดงตัวอย่างการใช้โครงข่ายประสาทเทียมในการแก้ปัญหาจริง

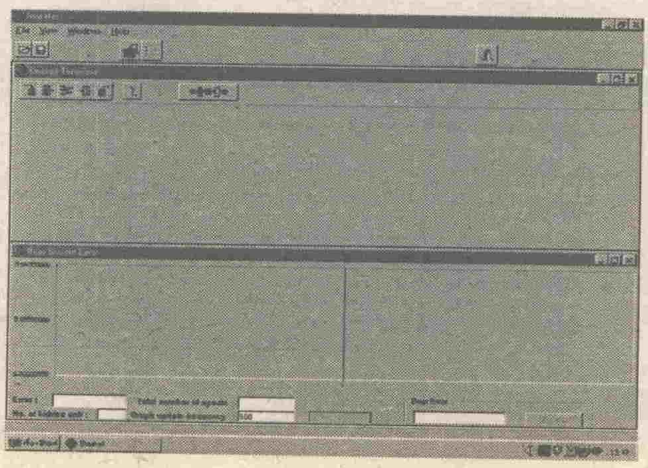
ซอฟต์แวร์นี้พร้อมทั้งคู่มือสามารถดาวน์โหลดได้ที่ http://www.s-t.au.ac.th/senior/index.html ให้หัวข้อโครงงานชื่อ Stat Net (Neural Networks for Statistical Modeling) ซอฟต์แวร์นี้ทำงานบนระบบปฏิบัติการ Windows 95 เป็นต้นไป ต้องการ RAM 16 MB และความละเอียดของจอภาพที่ 800x600 dpi ขึ้นไป ซอฟต์แวร์นี้ใช้ได้เฉพาะกับโครงข่ายประสาทเทียมแบบเพอร์เซพตรอนหลายชั้นกับกฎการเรียนรู้แบบแพร่ย้อนกลับเท่านั้น

เมื่อ install ซอฟต์แวร์และเปิดทำงานจะได้จอภาพแรกหน้าตาดังรูปที่ 7 ซึ่งแบ่งเป็น 2 หน้าต่างย่อยคือหน้าต่างสำหรับการออกแบบและประมวลผลโครงข่ายประสาทเทียม (Design Template) และหน้าต่างแสดงค่าผลรวมความผิดพลาดยกกำลังสอง (Sum Square Error) ซึ่งใช้แสดงค่าความผิดพลาดในขณะการฝึกหัดโครงข่าย ในการเริ่มใช้ซอฟต์แวร์นี้เราก็ต้องไปออกแบบโครงข่ายกันก่อนโดยใช้ Design Template ซึ่งจะเห็นได้ว่ามีไอคอนคำสั่งอยู่ทั้งหมด 7 คำสั่งเรียงจากซ้ายไปขวา ได้แก่ Input box, Preprocessor box, Neural network box, Postprocessor box, Output box, Connection line และ Automatic design

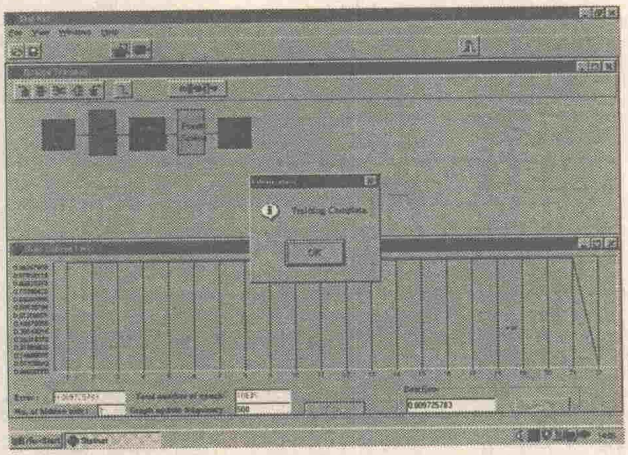
Input box ใช้สำหรับระบุอินพุตของโครงข่ายใหม่ที่จะสร้าง สามารถมีได้หลาย ๆ กล่อง ส่วน Preprocessor และ Postprocessor ใช้สำหรับการปรับสัดส่วนของข้อมูลอินพุตให้มีค่าอยู่ในช่วง [0, 1] และใช้เพิ่มสัดส่วนของเอาต์พุตเพื่อให้ได้ข้อมูลตามค่าความเป็นจริงตามลำดับ ส่วน Neural Network ใช้ในการสร้างโครงข่ายประสาทเทียมแบบ MLP และ Output ใช้ระบุเอาต์พุตของโครงข่าย กล่องเหล่านี้สร้างได้โดยกดคลิกที่รูปไอคอนนั้น ๆ แล้วกดอีกทีในบริเวณพื้นที่ว่างด้านล่างของไอคอนในหน้าต่าง Design Template หลังจากนั้นกดปุ่ม Connection line เพื่อใช้เชื่อมต่อกล่องต่าง ๆ เข้าด้วยกัน รูปแบบการเชื่อมต่อข้อมูลจะต้องเป็นในลักษณะนี้เท่านั้น

```
Input → Preprocessor → Neural
Network → Postprocessor → Output
```

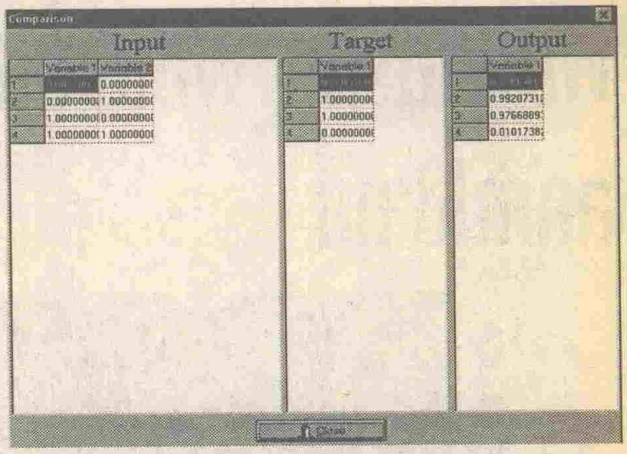
โดยที่ถ้าข้อมูลอินพุตอยู่ในช่วง [0, 1] อยู่แล้ว Preprocessor และ Postprocessor ก็ไม่จำเป็นต้องมี แถมทิปในการทำ connection line จากผู้พัฒนาซอฟต์แวร์นิดหนึ่งคือ ถ้าการต่อเชื่อมประสบผลสำเร็จ จุดที่เชื่อมต่อกับกล่องจะกลายเป็นสีแดง ถ้ายังเป็นสีดำอยู่แปลว่ายังต่อไม่สำเร็จ ให้ทำการขยับเส้นเชื่อมไปใกล้กล่องมากขึ้นจนกว่าจุดต่อเชื่อมจะกลายเป็นสีแดง เป็นอันว่าประสบผลสำเร็จ ถ้าการต่อเชื่อมไม่สมบูรณ์ เมื่อสั่งฝึกหัดโครงข่าย จะมี Error Message ขึ้นบอกว่า "Error : Connection not complete" หลังจากที่ต่อเชื่อมเรียบร้อยแล้ว ผู้ใช้สามารถตั้งค่าพารามิเตอร์ได้โดยการดับเบิลคลิกที่กล่องที่สร้างไว้ หรือกดเมนู View เลือกคำสั่งย่อย Setting ก็ได้เช่นกัน ถ้าท่านต้องการทดสอบโครงข่ายประสาทเทียมตัวอย่างที่ซอฟต์แวร์สร้างไว้ให้แล้ว หรือต้องการเลือกใช้โครงข่ายที่สร้างไว้แล้ว ท่านสามารถเลือกได้จากคำสั่ง Automatic Design เมื่อระบุข้อมูลอินพุตและเอาต์พุตเรียบร้อยแล้ว ผู้ใช้สามารถสั่งให้โครงข่ายฝึกหัดได้โดยใช้ไอคอนที่สาม สีน้ำเงินของหน้าต่างหลักได้ดังภาพในรูปที่ 7 ซึ่งเมื่อกดฝึกหัด หน้าต่างย่อยที่สอง Sum Square Error จะมีการแสดงผลค่าความผิดพลาดอยู่ตลอดเวลา กราฟจะเปลี่ยนแปลงจนกระทั่งความผิดพลาดลดลงถึงจุดที่ได้ระบุไว้ใน Neural network box ซึ่งในซอฟต์แวร์นี้จะใช้เทคนิคพรุนนิงช่วยในการหาจำนวนนิวรอนในชั้นซ่อนที่เหมาะสมให้ด้วย ซึ่งเมื่อการฝึกหัดเสร็จสิ้นลง จะปรากฏตัวเลขแสดงจำนวน Hiden neuron, จำนวนรอบทั้งหมดที่ใช้ในการฝึกหัด และค่าความผิดพลาดท้ายสุดที่ยอมรับได้



รูปที่ 7
แสดงหน้าต่างหลักของ Stat Net



รูปที่ 8 แสดงหน้าต่าง Design Template และ Sum Square Error ของปัญหา XOR



รูปที่ 9 แสดงหน้าต่างการเปรียบเทียบอินพุต เอาต์พุต และเป้าหมาย ที่ได้จากการฝึกหัดปัญหา XOR

ผู้เขียนขอเสนอตัวอย่างการใช้ซอฟต์แวร์นี้กับปัญหา XOR ที่ได้กล่าวไปแล้วในตอนต้น โดยจะกำหนดค่าอินพุต "จริง" เป็น 1 และ "เท็จ" เป็น 0 ซึ่งโครงข่ายสำหรับปัญหานี้สามารถเลือกได้จากปุ่ม Automatic design ได้เลย หรือผู้อ่านจะลองสร้างเองก็ได้เช่นกัน รูปแสดงโครงข่ายประสาทเทียมพร้อมกราฟแสดงค่าความผิดพลาดแสดงในรูปที่ 8 ส่วนค่าอินพุต เอาต์พุต และเป้าหมาย (Target) นั้นสามารถเรียกดูได้โดยเมนู คำสั่ง View คำสั่งย่อย Comparison ผลที่ได้ดังแสดงในรูปที่ 9 ซึ่งจะเห็นได้ว่าค่าเป้าหมายและค่าเอาต์พุตมีค่าใกล้เคียงกันมาก นั่นคือผลลัพธ์ถูกต้องอยู่ในระดับค่าความผิดพลาดที่กำหนดไว้แล้ว (0.001)

ในฉบับหน้าจะกล่าวถึงแบบจำลองสำหรับการใช้งานแบบเกี่ยวข้องกัน (Association) และแบบจัดตัวเอง (Self-Organization) และการประยุกต์ใช้แบบจำลองเพื่อใช้กับงานหาค่าเหมาะสม (Optimization) เป็นการสรุปบทความนี้ สำหรับซอฟต์แวร์ที่ทำหน้าที่ตัวจำลอง (Simulator) ของโครงข่ายประสาทเทียมในปัจจุบันมีอยู่เยอะมาก ไม่ว่าจะเป็นซอฟต์แวร์แบบ Commercial หรือซอฟต์แวร์แบบ Shareware และ Freeware สนใจหาข้อมูลเพิ่มเติมได้ที่ http://www.emsl.pnl.gov:2080/proj/neuron/neural/systems/ ซึ่งได้ทำการรวบรวมลิงก์สำหรับซอฟต์แวร์ต่าง ๆ ไว้มากมาย

พบกันใหม่ในตอนหน้า สวัสดี

หนังสืออ้างอิง

[1] Limin Fu, "Neural Networks in Computer Intelligence", MIT Press and McGraw-Hill Inc., 1994.

[2] Laurene Fausett, "Fundamentals of Neural Networks : Architectures, Algorithms, and Applications", Prentice Hall International Editions, 1994.

[3] Simon Haykin, "Neural Networks - A Comprehensive Foundation", Macmillan College Publishing Company, 1994.

[4] S. Y. Kung, "Digital Neural Networks", Prentice Hall, 1993.

[5] C. Lursinsap and C. Tanprasert (Khunasaraphan), "Simulated Light Sensitive Model for Handwritten Digit Recognition", International Joint Conference on Neural Networks, Baltimore, Maryland, pp. IV-13-IV-18, June 1992.

[6] C. H. Sequin and R. R. Clay, "Fault Tolerance in Artificial neural Networks", International Joint Conference on Neural Networks, pp. I-703-708, 1990.

[7] คู่มือการใช้ซอฟต์แวร์ Stat Net จาก http://www.s-t.au.ac.th/senior/index.html